{EPITECH}

JOB BOARD_

< WEB PROJECT />

JOB BOARD

This project aims to realize the following:

- ✓ a database to store job advertisements ;
- ✓ a webpage (*front end*) using Javascript technologies to display an online job advertisements board as well as an administration page for the admin user ;
- ✓ an API (*back end*):
  - to allow users to apply for jobs ;
  - to manage the DB (only for admin user) with CRUD operations.

⚠️ Mind out! Each step depends on the previous one...

📢 No need to create different folders for rendering each step. These are there to help you organize yourself on the project!
You must detail and attach the elements concerning these steps in the Readme which you will add to the rendering directory.

All your files must be tested on your local host.

Step 01

Create an SQL database to store job advertisements.
It must contain at least:

- ✓ a table to store advertisements ;
- ✓ a table to store companies ;
- ✓ a table to store people (in charge of or applying to an advertisement) ;
- ✓ a table to keep information about a job application (emails sent, people/ad concerned, etc.).

You can add as many relevant tables/fields as you want.

Step 02

Write an HTML/CSS page showing several job advertisements. For each ad, there must be at least a place for a title, one for a short description, and a ”learn more” button.

Clicking on this ”learn more” button has no action for the moment.

The design of this page is up to you, but do not spend too much time on it now.

Step 03

The ”learn more” button must now display all the available information about the ad (full description, wages, place, working time,...), without reloading the page.
No popup.

Keep the database fields coherent with the information you display on the HTML page.

Once again, the design is up to you: put this data wherever you like.

Step 04

Create an API providing CRUD operations on the database tables (Create, Read, Update, Delete).
The ”learn more” button must be linked to an API route to fetch the job information dynamically.

Attention must be paid to using the appropriate HTTP verbs and the restful API routing rules

You can use the language you feel comfortable with. Here are some candidates: Python (with Flask or FastAPI), NodeJS (with Express.js), PHP (with Laravel or Lumen), ...

Fill in the database using your backend.

Step 05

On your webpage, add an ”Apply” button for each ad.
When this button is clicked, it opens a form to

- ✓ enter information about you (name, email, phone,...)
- ✓ send a message to the owner of the ad (you).

This action must be saved in the database.

Step 06

Add an authentication mechanism on the webpage.
When identified, you don't have to fill in your personal information to apply to a job.

You need a **connection** page and another one to create/modify an account.

Step 07

Create an HTML/CSS page for monitoring the database. From this page the user can list all the records of your tables, can create new records and can update or delete the existing ones.

This page can be accessed only by an admin. So a successful connection redirects to this page for an admin user or to the page created at step 02 otherwise.

Think about pagination when displaying a big list of database records

The conception and page layout is up to you. Make them simple and user-friendly

Step 08

Polish your pages up by improving their design, tweaking and refining the style sheets.

v 3.5.1

{EPITECH}